The 14th ACM Workshop on Hot Topics in Storage and File Systems,
Virtual, June. 27~28, 2022

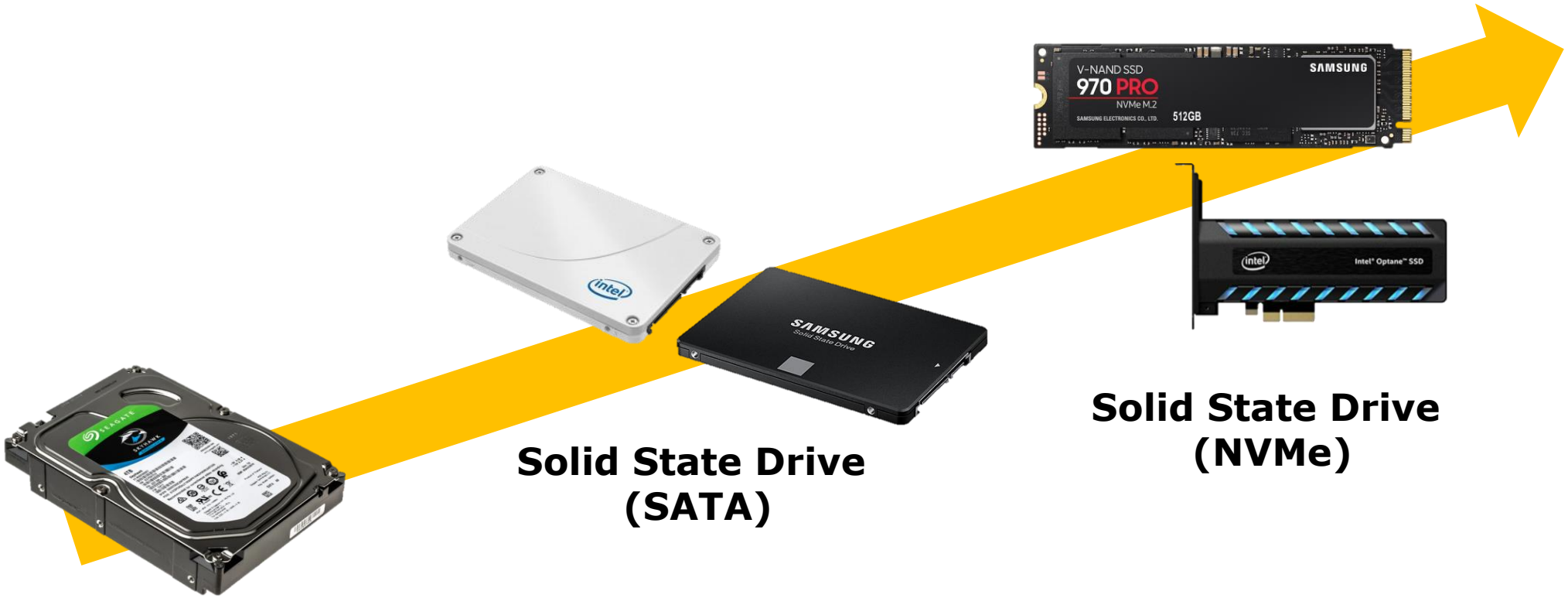
Fair I/O Scheduler for Alleviating Read/Write Interference by Forced Unit Access in Flash Memory

Jieun Kim Dohyun Kim Youjip Won

Department of Electrical Engineering,

Introduction

Storage Trends



Hard Disk Drive

Solid State Drive (SATA)

Solid State Drive (NVMe)

> 1 *M* IOPS

Sub-msec latency

I/O Interference

Multiple tenants share the same storage device.

Challenge

Alleviating I/O Interference among the multiple tenants.

FUA Interference

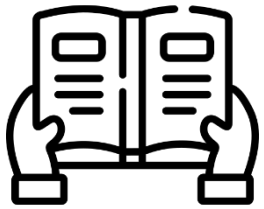
- Forced Unit Access (FUA) is a flag set for the I/O command.
  - FUA enforces the data to directly reflect to the disk.
  - QEMU-KVM directsync cache mode & Microsoft SQL server

- Read/Write interference in flash chip
  - Read is postponed by the preceding write in the same die.
  - Write latency in flash chip: 10-40 times longer than read.

- **FUA Interference**
  - The FUA-flagged write is highly related to the read/write interference
  - The FUA-flagged write requests from host can occur the read/write interference.

Motivation

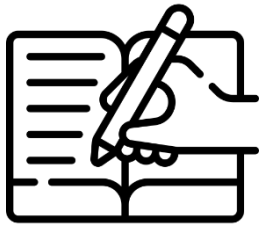
Simulation of FUA Interference



Reader

Random Read

Direct & Sync (`fio`)



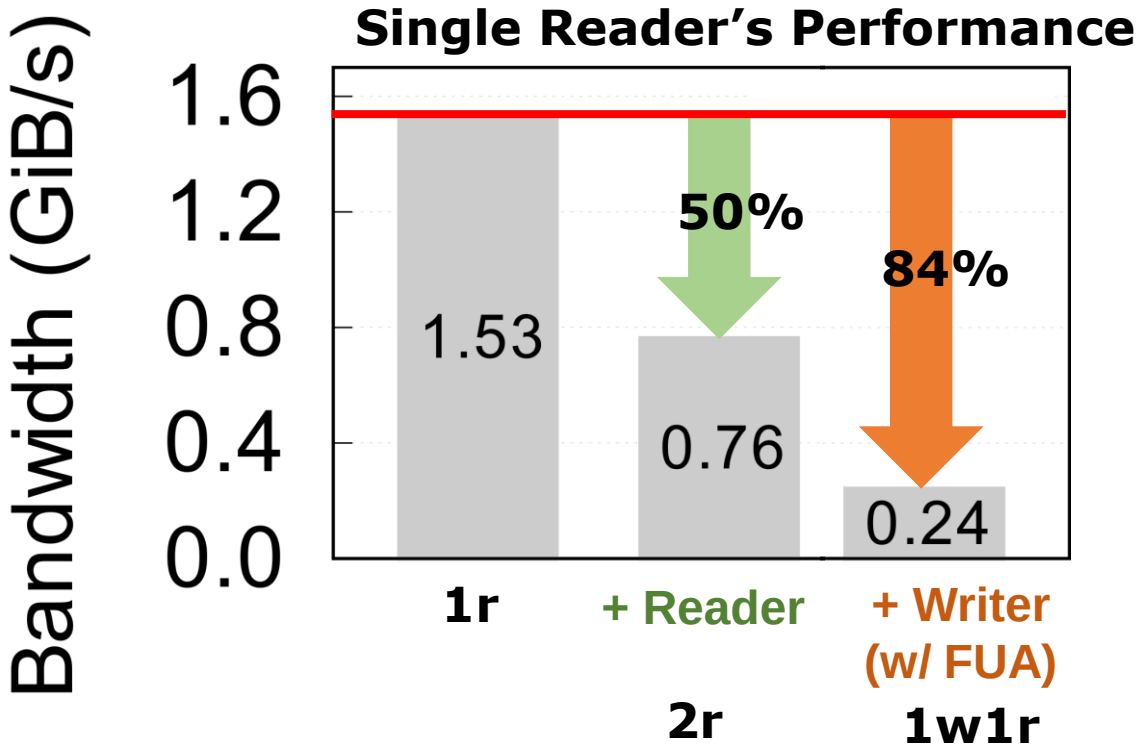
Writer

Random Write

Direct & Sync (`fio`)

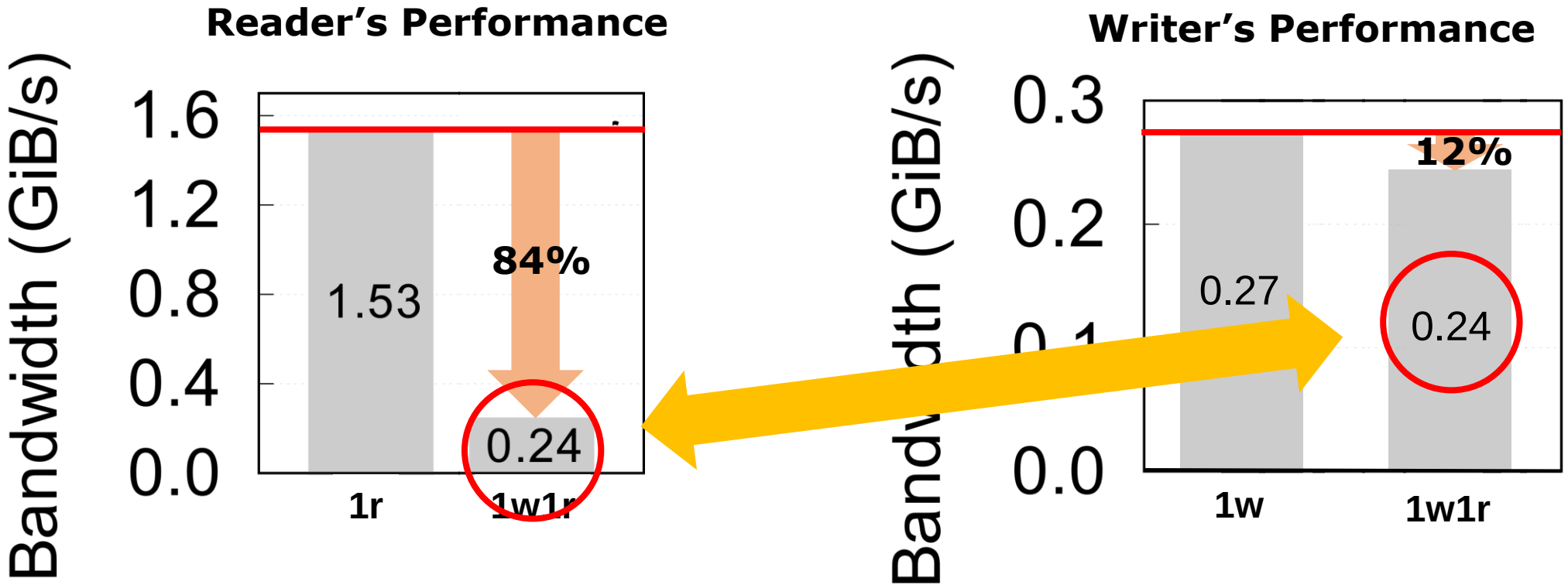
- Samsung 970 Pro 512 Gbyte, NVMe SSD.
- `fio` benchmark: "Direct & Sync" options
  - All the write requests are dispatched with the FUA flag.
- Workload Name: $\{N_W\}$w$\{N_R\}$r
  - N_W: The number of writers / N_R: The number of readers
  - ex) Single Reader (1r)
  - ex) Two readers & one writer (1w2r)

FUA Interference Analysis



- According to I/O type of concurrent worker, the bandwidth significantly varies.
- FUA-flagged write drops the reader's performance mostly.

Unfairness

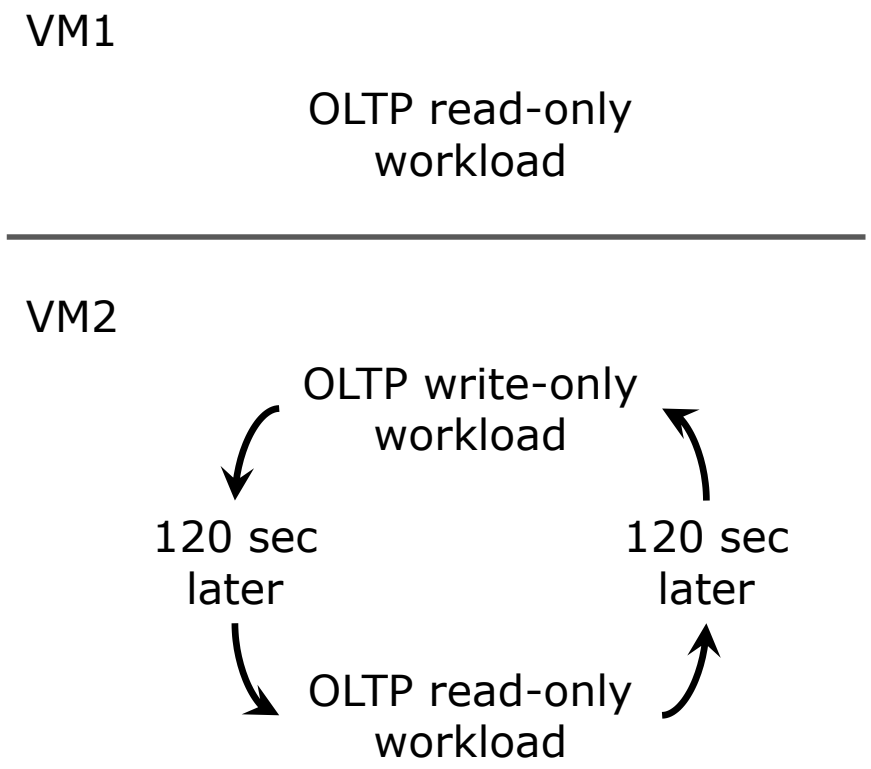


Unfairness between the read & FUA-flagged write.

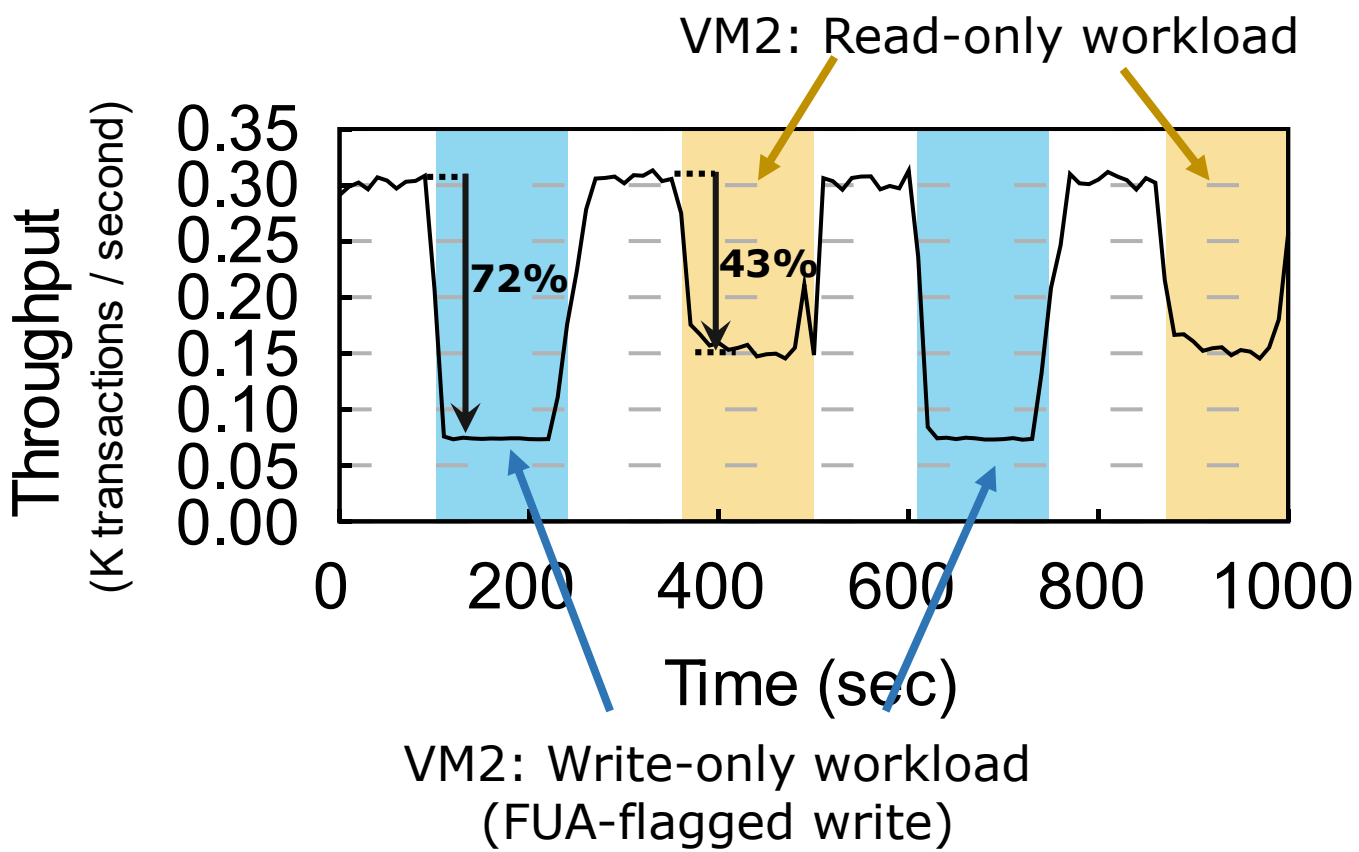
Real-World Workload

QEMU-KVM directsync cache mode

All the write requests are flagged with FUA.



VM1's Performance



Fair I/O Scheduling

- CFQ (Completely Fair Queueing)
- BFQ (Budget Fair Queueing)
- Deadline scheduler

→ *Provide fair I/O performance **among processes.***

☹ They do not consider the different I/O types.

- FIOS (FAST'12)

→ *Resolve **read/write interference** in host level.*

☹ It just blocks all write requests during dispatching the read requests.

- P/E Suspension (FAST'12)

→ *Resolve **read/write interference** in firmware level.*

☹ User cannot choose whether to adopt or not.

TABS per-Type fAir Bandwidth I/O Scheduler

Overview

- **Fairness Goal**
  - For each I/O type,

    (1) Bandwidth ∝ Maximum Bandwidth

    (2) Bandwidth ∝ Dispatch Rate

- **Key Idea**
  - Control dispatch rate of the FUA-flagged write to achieve the fair bandwidth.

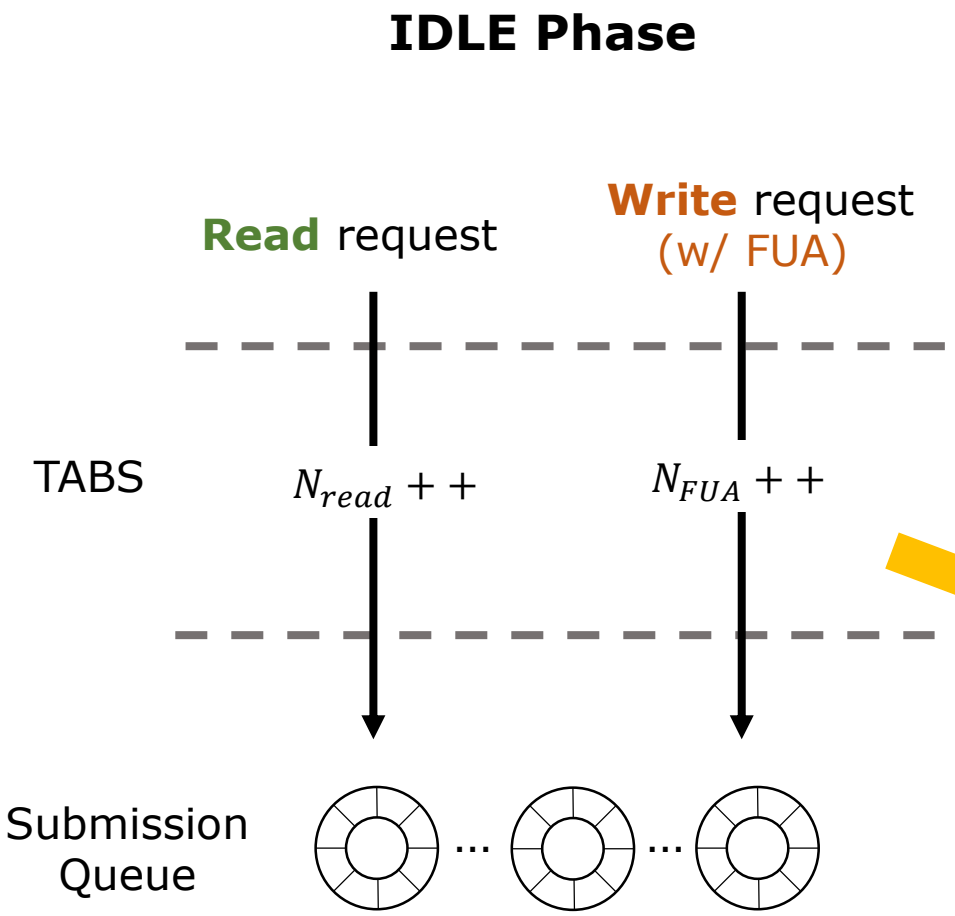
    (1) Two-phase Dynamic Scheduling

    ➔ To respond the various workloads.

    (2) Software-based Feedback

    ➔ For more precise scheduling according to the environment.

Two-phase Dynamic Scheduling



N: # of dispatched I/Os

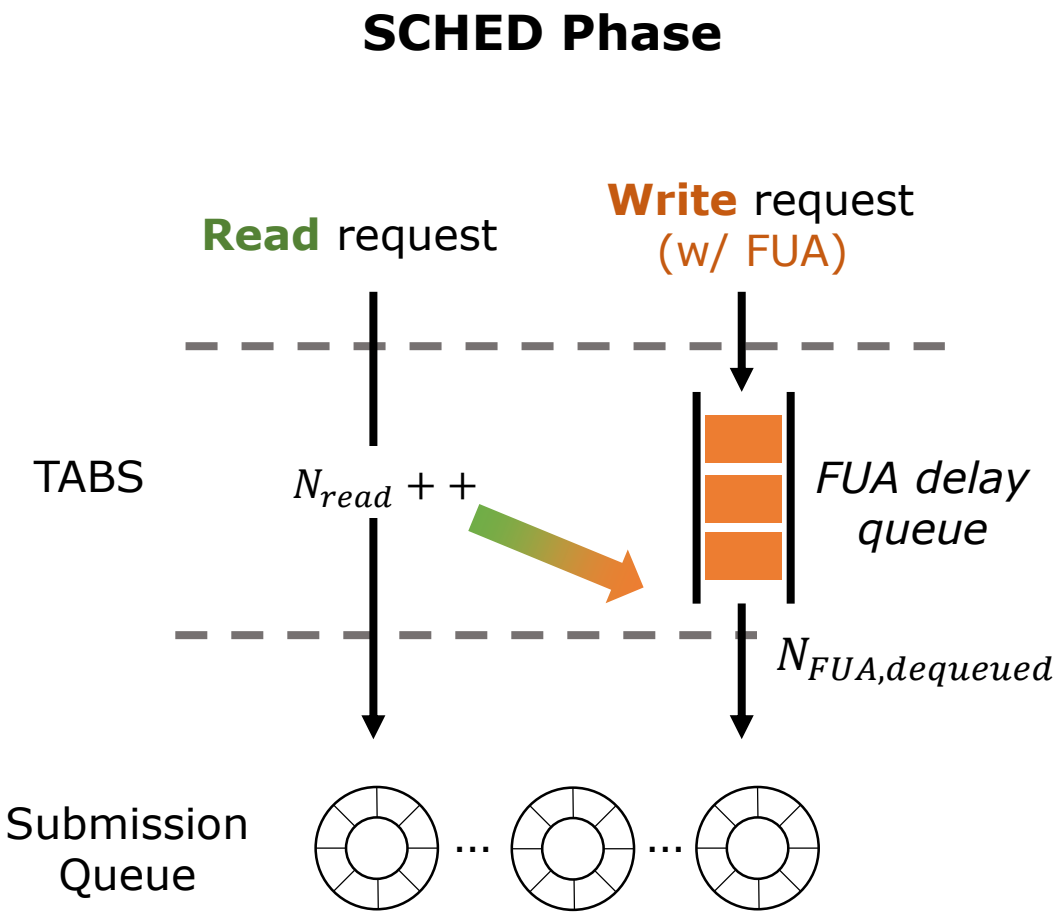
- IDLE Phase
  - TABS observes the I/O pattern of current workload.
  - Scheduling is disabled.
  - TABS counts the number of dispatched I/O requests for each I/O type (read & FUA write).
  - Phase is switched to SCHED when total I/O dispatch rate exceeds the threshold.

Fairness Goal: $Fair.BW$

$$Fair.BW_{read} = Max.BW_{read} \times \frac{N_{read}}{N_{read} + N_{FUA}}$$

$$Fair.BW_{FUA} = Max.BW_{FUA} \times \frac{N_{FUA}}{N_{read} + N_{FUA}}$$

Two-phase Dynamic Scheduling



- SCHED Phase
  - TABS inserts the FUA-flagged write requests into *FUA delay queue*.
  - TABS only dispatches the FUA writes as many as $N_{FUA,dequeued}$.
  - Throttling factor is the main factor that determines the I/O performance.

$$Throttling\ Factor = N_{read}\ /N_{FUA}$$

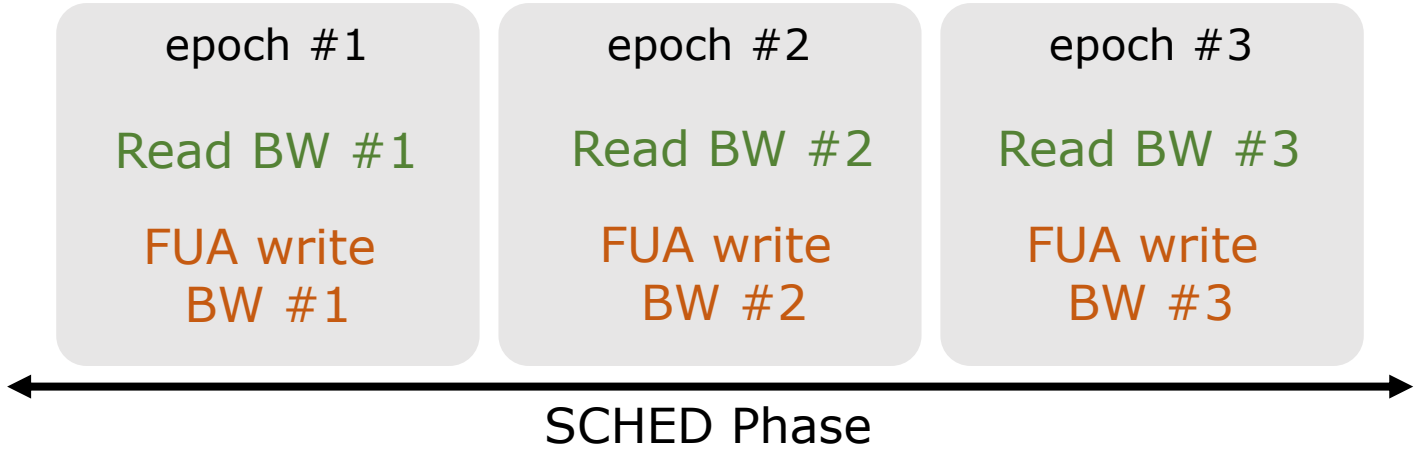
$$N_{FUA,dequeued} = Throttling\ Factor \times N_{read}$$

N: # of dispatched I/Os

Software-based Feedback

- TABS periodically adjusts the throttling factor to guarantee the fairness more precisely.

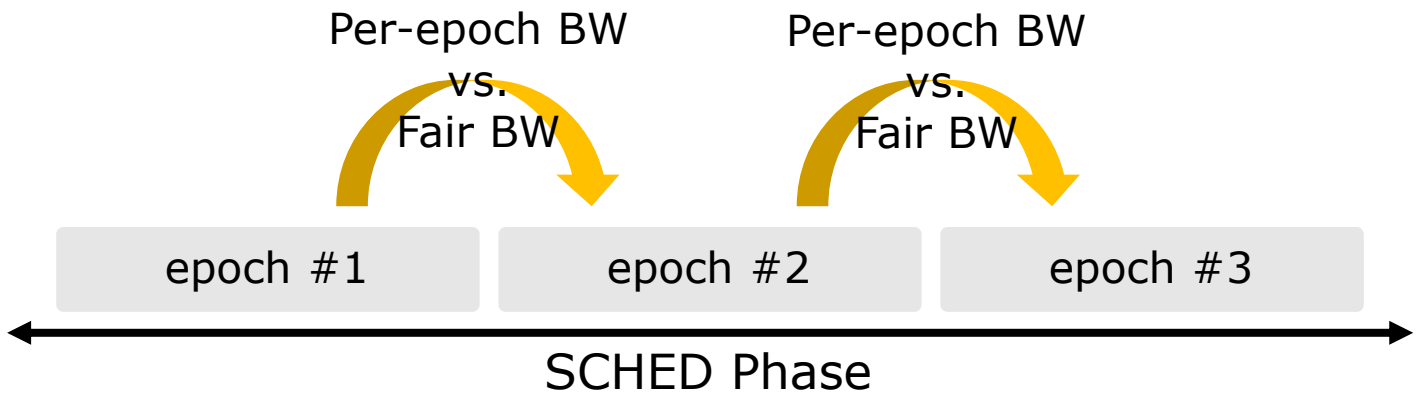
- TABS divides the SCHED phase into the same time interval, *"epoch"*.
- TABS keeps monitoring **the per-epoch bandwidth of each I/O type.**



Software-based Feedback

- For every end of the epoch, TABS performs the feedback.
  - For the each type of the I/O, compares **per-epoch bandwidth** and **fair bandwidth**.
  - Adjusts the throttling factor to minimize the gap between the two bandwidths.

For the BW of FUA-flagged write,	For the BW of read,
$Throttling\ Factor\ \uparrow$	$Throttling\ Factor\ \downarrow$



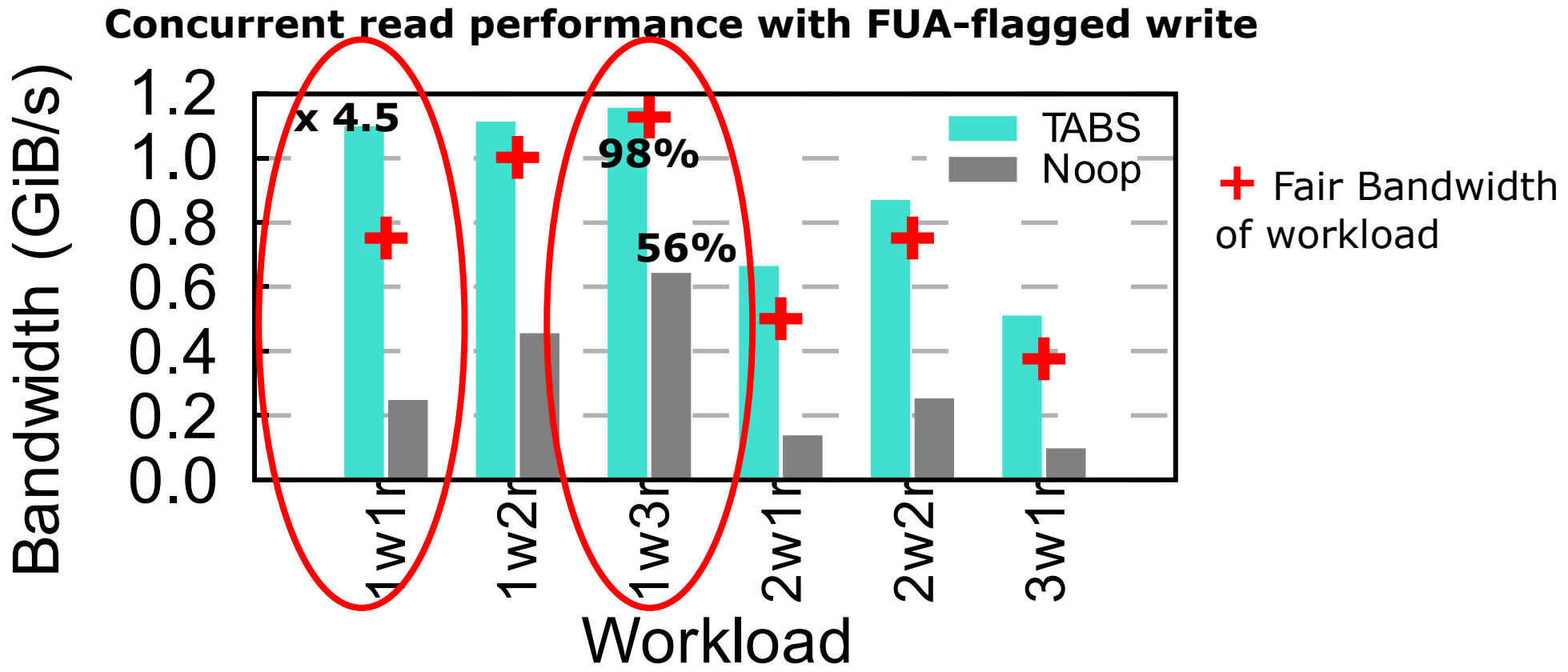
Evaluation

Environment

- Hardware Spec.
  - 4 Cores / 8 Threads (Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz)
  - 16 Gbyte DRAM
  - Samsung 970 Pro (512 Gbyte) NVMe SSD

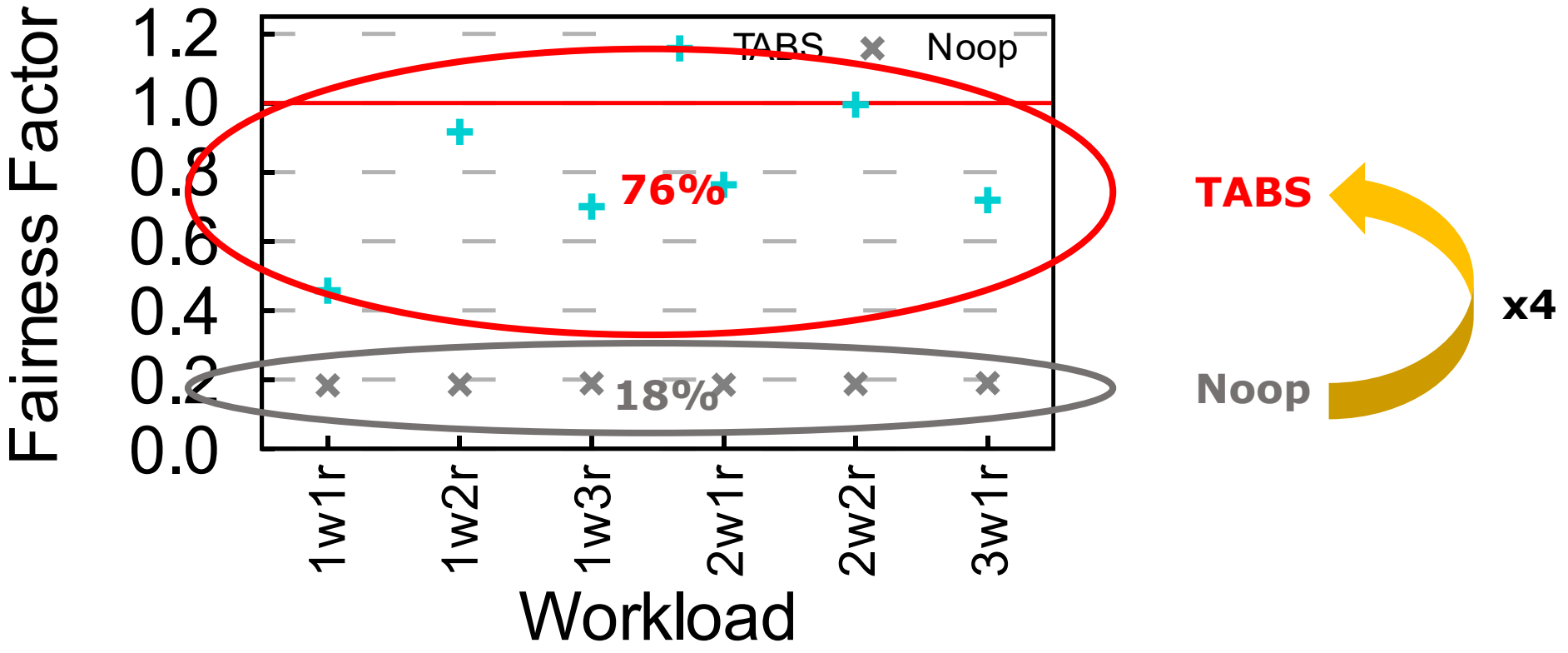
- Software Configurations
  - Centos 7.6 (Linux v5.8.5)
  - **`fio`** (version 3.7)
  - Compared with the **Noop** scheduler

Fairness Evaluation



- The read bandwidth is more closer to the fair bandwidth in TABS than Noop.
- TABS gives the device the possibility to accommodate more I/O traffic.

Fairness Factor



- Fairness factor represents how fair the read and FUA-flagged write are in terms of *bandwidth*.
- The fairness factor is close to 1 ➔ Read/FUA-flagged write requests become fairer.

Conclusion

Conclusion

- **FUA interference** is the cause of the unfairness between read and FUA-flagged write.

- TABS provides the fairness between **the different types of I/Os**.

  (1) Bandwidth ∝ Maximum Bandwidth

  (2) Bandwidth ∝ Dispatch Rate

- To achieve the fair bandwidth, TABS **throttles the FUA-flagged write requests**.

Two-phase Dynamic Scheduling

To respond the various workloads.

Software-based Feedback

For more precise scheduling according to the environment.

- TABS achieves **4 times higher fairness** than the Noop scheduler.

Thank you

Email:
pipiato@kaist.ac.kr Jieun Kim
ehgus421210@kaist.ac.kr Dohyun Kim